Anding, Katharina; Kuritcyn, Petr; Garten, Daniel

Using artificial intelligence strategies for process-related automated inspection in the production environment

Original published in:

Journal of physics / Conference Series. - Bristol : IOP Publ. - 772 (2016), 1, art. 012026, 6 pp.

ISSN (online): 1742-6596
ISSN (print): 1742-6588
DOI: 10.1088/1742-6596/772/1/012026
URL: http://iopscience.iop.org/1742-6596/772/1/012026
[*Visited:* 2017-10-19]

Using artificial intelligence strategies for process-related automated inspection in the production environment

K Anding[1], P Kuritcyn[1] and D Garten[2]

[1]Ilmenau University of Technology, Gustav-Kirchhoff-Platz 2, 98693 Ilmenau, Germany
[2]GFE Schmalkalden e.V., Näherstiller Straße 10, 98574 Schmalkalden, Germany

katharina.anding@tu-ilmenau.de, petr.kuritcyn@tu-ilmenau.de, d.garten@gfe-net.de

Abstract. In this paper a new method for the automatic visual inspection of metallic surfaces is proposed by using Convolutional Neural Networks (CNN). The different combinations of network parameters were developed and tested. The obtained results of CNN were analysed and compared with the results of our previous investigations with color and texture features as input parameters for a Support Vector Machine. Advantages and disadvantages of the different classifying methods are explained.

1. Introduction

Lately, biologically inspired CNN are becoming more popular in the area of computer vision. The CNN is considered a special case of standard neural networks. They showed exceptional performance for various tasks as image recognition, object detection [1], feature extraction [2] and segmentation [3]. Since the first seminal paper [4], convolutional nets are continuously advancing their accuracy in different image classification tasks [5] and proved robustness against different lighting conditions, object positions, partial occlusions and translation. Despite many of the recent CNNs are very time-consuming in both training and testing, they have an important advantage in comparison to sophisticated methods in machine learning (like Support Vector Machine, decision trees etc.): CNNs usually do not need to use both external feature extraction and feature selection methods for image recognition tasks. Low-level image features like edges, lines, and corners are extracted out of the convolution layers. In the following layers, patterns are detected in these low-level features. Therefore, we get another level of abstraction within every layer and the concept can be classified among the group of deep learning methods [6].

In this paper, we propose a new method for the automatic visual inspection of metallic surfaces. We have presented a method for recognition of metal surfaces defects based on the analysis of drilled countersink holes in heat-treatable steel [7]. The classification was done using color and texture features as input parameters for a Support Vector Machine. Here we want to present results from our latest research considering the usage of Convolutional Neural Networks (CNN) for this task. We discriminated between the two classes: good parts (perfect surface) and bad parts (longitudinal rills and chatter marks) (see Figure 1).

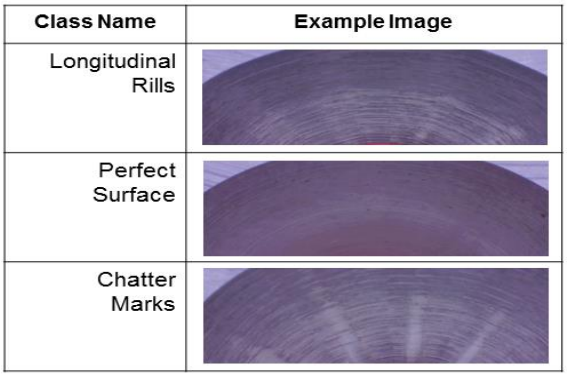

Figure 1. Perfect surface and defect samples

Firstly, we will describe the design of our approach, then present our results and compare with our previous investigations at the end.

2. Development of a novel cnn recognition approach for metal surfaces defects

2.1. Data preprocessing

Our dataset consists of 3 classes (see Figure 1) and around 250 RGB-images of resolution 1280x1024 pixels. That is a huge size for the input of convolutional network (it requires more than 11 million parameters only for one filter with size 3x3). We needed to downsample images and to reduce the number of color layers from 3 to 1 (grayscale). It does not change drastically the difficulty of detection, but improves the calculation speed and reduces the number of parameters in the net (16 times less for the first convolution layer). We obtained gray-scaled images of resolution 320x256 pixels after downsampling with antialiasing filter (package Pillow [8]). Two datasets were prepared: with two classes (good parts – bad parts) and with three classes (perfect surface - longitudinal rills - chatter marks).

2.2. Design of the network

Following packages were used for building our convolutional neuronal network: Theano [9], Lasagne [10] and nolearn[11]. The implementation consists of many elements of CNN: convolutional layers, nonlinearity functions, subsampling layers etc.

The structure of network was inspired by LeNet network [4]. Figure 2 shows the network structure. It consists of several types of layers: convolutional, pooling and dense layer. We tested different combinations of these elements with different parameters.

Three pooling layers perform downsampling after each convolutional layer. After all of them we placed two fully-connected dense (hidden) layers and then an output layer.

Each convolutional layer has a nonlinearity function placed directly after them. We tried several nonlinearity functions, which were successful in other investigations: rectifier linear unit (ReLU)[12], exponential linear unit (ELU)[13], parametric rectifier linear unit [14].

After output layer, we placed the softmax function [15], as a typical choice for the classification task, which generalizes the categorical probability distribution

For the parameter updates, we used the algorithm AdaMax [16]. This method worked well in our investigations and showed a good convergence and optimization properties in comparison to other stochastic optimization methods (Stochastic Gradient Descent (SGD)[17], Nesterov Momentum [18]).

All computations were performed on the single GPU-unit.

2.3. Data augmentation

The first problem we have faced in our investigations with CNN was a small size of the given dataset. Usually, neural networks require big datasets to achieve good results [5]. There are many techniques to make a dataset bigger without necessity of new experiments. The general term for that process is data augmentation. At first, we used flipping randomly chosen images. In the next stage we added also random rotation (0, 90, 180, 270 degrees), horizontal and vertical shift for a few pixels and zooming in the range between 0.9 and 1.1 of the original image size.
For training our net, we used small batches from the original dataset. These batches were augmented on the fly by an implemented generator. It generates new samples at "no cost".

2.4. Prevention of the overfitting

Usual problem when using small datasets and feedforward networks is the overfitting effect [5]. To prevent this, we applied some regularization techniques: batch normalization [19] and dropout [20]. Batch normalization is a normalization step, which fixes variances and means of layer inputs. It improves the stability and speed of the optimization process. The idea of dropout is to randomly nullify the weights of neurons from the previous layer. It prevents adaptation of neurons to specific features in images and improves the generalization ability of the network.

3. Experimental results

The results of our investigations are presented in Table 1.

There are a few most successful network configurations with the highest accuracy for the given dataset. The networks consist of a few stages with pooling layers between them. Each stage consists of one or several convolutional layers. The first number in brackets corresponds to the size of the filter, the second to the number of filters with that size. Slash sign shows stride of the filter or pooling layer. The filters on each stage are working consecutively from top to bottom, not in parallel. The networks A-D were tested on the dataset with downsampled images. The network E was tested on the dataset with full-sized grayscale images.

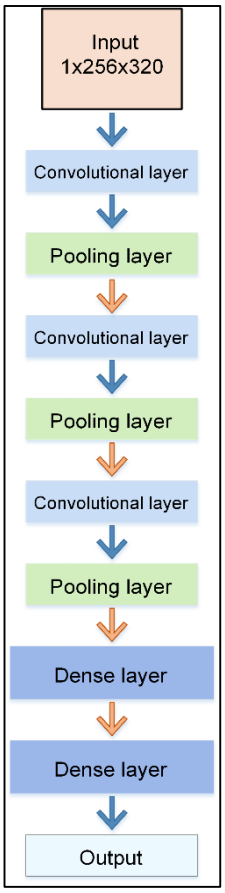

Figure 2. Structure of neuronal network for inspection of metal surfaces

Table 1. Configuration and performance of several convolutional neuronal networks on the metal surface dataset

	Stage 1	Pool	Stage 2	Pool	Stage 3	Pool	Stage 4	Pool	Fully-connected layer	Total recognition rate		Nonlinearity function	Training time (sec/epoch)
										2 class	3 class		
A	(12, 32)	2/2 Drop. 10%	(6, 64)	2/2 Drop. 20%	(6, 128)	2/2 Drop. 30%			2x256 Drop. 30%	84.7	56.4	ELU	4.75
B	(3, 16) (3, 32)	2/2	(3, 64)	2/2 Drop. 10%					2x256 Drop. 10%	96.5	83.5	ELU	13
C	(3, 8) (3, 16)	2/2	(3, 16) (3, 32) (3, 64)	2/2	(3, 64)	2/2	(3, 128) (3, 128)	2/2 Drop. 10%	2x256 Drop. 30%	96.5	75.3	ELU	10
D	(3, 16) (3, 32)	2/2	(3, 64) (3, 128)	2/2	(3, 256)	2/2	(3, 512)	2/2 Drop. 10%	2x256 Drop. 30%	96.5	81.2	ELU	19.5
E	(11, 1)/4 (3, 16) (32, 3)	2/2	(3, 64) (3, 128)	2/2	(3, 256)	2/2	(3, 512)	2/2 Drop. 10%	2x256 Drop. 30%	95.3	75.3	ELU	21

Our first attempts showed a recognition rate of 48%. It is a decent performance and we used data augmentation and dropout to improve the results. It is worth to notice that we tried to apply batch normalization technique, but it does not improve the performance of our networks. The network A uses a relatively big filter sizes and consists of only three convolutional layers. A recognition rate of 56.4% for the 3-class dataset and 84.7% for the 2-class dataset were achieved. As showed in [21], the results can be improved by reducing the filter size and increasing the depth of network and the number of filters. The network B used two consecutive convolutional layers with small filters on the input. This stack makes possible to develop the complex features. It results in increase of the total recognition rate up to 83.5% for 3-class dataset and to 96.5% for 2-class dataset. The networks C and D use more convolutional layers. The network C consists of fewer number of filters in the first two stages that reduces the computational time, but costs a few percent of performance on the 3-class dataset (8% less in comparison to B). The network D has additional layers on stages 2, 3 and 4 in comparison to B. It results in increasing the recognition rate of objects with longitudinal rills, but also decreases the recognition rate of objects with chatter marks. In general, the network D shows a comparable accuracy with network B, but takes 50% more time for training. It also shows a good feature generalization (see Figure 3). The salience plot shows the image parts, which matter to the net.

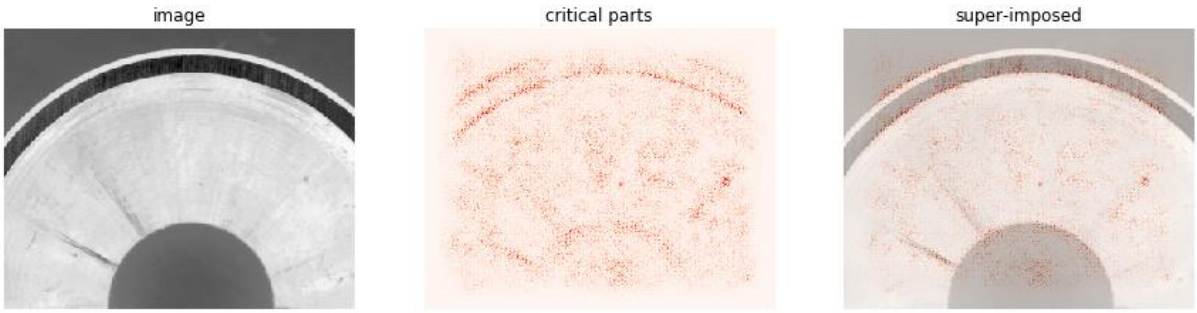

Figure 3. Salience plot of image with chatter marks (network D). (From left to right: original image, important parts of the image for classification, combination of those two images.)

The network E was designed for the dataset with full-sized images. It has the same structure as D, except the additional layer with filter size 11x11 and stride 4 on the input. This configuration was

successfully used in [5] and helps to downsize the image with extracting some information in comparison to other dataset with preprocessed images. It results in a high recognition rate both on the 3-class (75.3%) and on the 2-class (95.3%) datasets. However, this did not overperform the networks B and D. The network E also requires for 7% more time than D and 38% than B.

All of our most efficient networks are using exponential linear unit as nonlinearity function after convolution layers. The tests with other nonlinearity functions (rectifier linear unit and parametric rectifier linear unit) showed no improvements of recognition rate. It can be explained with the high dependency of these methods from the initializations weight in layers.

4. Conclusions

The total recognition rate of 96.5% was achieved by using a relatively small dataset and a not fully optimized convolutional neural network. Using full sized images does not improve the classification accuracy, but it increases the computation time depending on the network architecture (not less than 5%). Using small filter sizes and deeper networks results in increasing of the classification accuracy and improving the generalization ability of the network. Applying dropout to fully connected layers helps to prevent overfitting. An optimal value of 30% in our case. Exponential linear unit as nonlinearity function and AdaMax algorithm for parameter updates showed a good synergy in our tests.

An unoptimized Support Vector Machine (SVM) reached recognition rates of about 85%. With SVM-parameter-optimization and further feature developing a total recognition rate of 99% were achieved with an SVM-classifier. The great advantages of CNN over the SVM are:

- can be efficiently implemented with field programmable gate arrays (FPGA) and graphic processing units (GPU)
- convolution kernels with integral effects can reduce image noise
- feature extraction and classification are implemented in a single process
  ⇨ no need for time-consuming feature engineering and feature selection

A comparison between both algorithms is given in Table 2.

Table 2. Comparison between svm and convolutional neural networks.

Criteria	**svm**	**convolutional neural networks**
separate feature extraction	yes	no
separate feature selection	case-specific	no
prone to overfitting	no	with small datasets
parameter optimization needed	yes	yes

With these advantages of CNN it is possible to build classification systems in an automated way. A convolutional neural network will obtain sample images from the classes to discriminate between and a genetic algorithm will optimize the network architecture. With this optimized architecture the final network is trained.

Acknowledgments

The research project, which forms the basis of this paper, is supported by the Thuringian Ministry of Economy, Employment and Technology (TMWAT) with means from the European Social Fund (ESF). The responsibility for the content of this paper lies with the author.

The samples of metal surfaces with different surface defects, like scratches and chatter marks were produced by our project partner the society for production engineering and development (GFE e.V.) in Schmalkalden (Germany). Our special thanks goes to Dr. Daniel Garten and his colleagues for the production of metal surfaces and the image acquisition for building the dataset of metal surfaces with and without defects.

References

[1] Glorot X and Bengio Y. 2010 Understanding the difficulty of training deep feedforward neural networks *International Conference on Artificial Intelligence and Statistics* **249–256**, 2010
[2] Razavian A S, Azizpour H, Sullivan J, and Carlsson S 2014 Cnn features off-the-shelf: An astounding baseline for recogniton *CVPR 2014, DeepVision Workshop*
[3] Hariharan B, Arbel´aez P, Girshick R, and Malik J 2014 Simultaneous detection and segmentation *ECCV* **297–312**
[4] LeCun Y, Bottou L, Bengio Y, and Haffner P 1998 Gradient-based learning applied to document recognition *Proceedings of the IEEE, 86(11)* **2278–2324**
[5] Krizhevsky A, Sutskever I, Hinton G 2012 Imagenet classification with deep convolutional neural networks *Advances in Neural Information Processing Systems*
[6] Hijaz S e. a. 2016 Using Convolutional Neural Networks for Image Recognition *http://ip.cadence.com/uploads/901/cnn_wp-pdf*
[7] Garten D, Lahmann H W, Polte G and Anding K 2015 Photonic in-process quality measurements of surfaces and geometrical parameters of parts *MAZeT JENCOLOR-ZEISS SPECTROSCOPY-SENSORIK BAYERN Spectro-Net Cross-clustering Collaboration Forum, Jena http://spectronet.de/de/videos_2015/ video-photonic-in-process-quality-measurements-of-_ie2nz5z9.html*
[8] Git-Repository with source code and documentation of Pillow package *https://github.com/python-pillow/Pillow*
[9] Theano 0.8.0 documentation *http://deeplearning.net/software/theano*
[10] Git-Repository with source code and documentation of Lasagne-Framework *https://github.com/Lasagne/Lasagne*
[11] Git-Repository with source code and documentation of nolearn-Framework *https://github.com/dnouri/nolearn*
[12] Maas et al. 2013 Rectifier Nonlinearities Improve Neural Network Acoustic Models *http://web.stanford.edu/~awni/papers/relu_hybrid_icml2013_final.pdf*
[13] Clevert D A, Unterthiner T, Hochreiter S 2015 Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs) *http://arxiv.org/abs/1511.07289*
[14] He K, Zhang X et al. 2015 Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification *http://arxiv.org/abs/1502.01852*
[15] Bishop C M 2006 Pattern Recognition and Machine Learning *Springer*
[16] Diederik K and Ba J 2014 Adam: A Method for Stochastic Optimization *arXiv preprint arXiv:1412.6980*
[17] LeCun Y, Boser B, Denker J S, Henderson D, Howard R E, Hubbard W and Jackel L D 1989 Backpropagation applied to handwritten zip code recognition *Neural computation*
[18] Nesterov Y 1983 A method of solving a convex programming problem with convergence rate O(1/k2) *In Soviet Mathematics Doklady, volume 27*, **372–376**
[19] Ioffe S and Szegedy C 2015 Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift *http://arxiv.org/abs/1502.03167*
[20] Hinton G, Srivastava N, Krizhevsky A, Sutskever I and Salakhutdinov R 2012 Improving neural networks by preventing co-adaptation of feature detector *arXiv preprint arXiv:1207.0580*
[21] He K, Sun J 2014 Convolutional Neural Networks at Constrained Time Cost *http://arxiv.org/abs/1412.1710v1*